. **Briefly explain types of JDBC drives.** **(05 Marks)**
Depending on the relative location of the data source and the application, several architectural scenarios are possible. For example, drivers in JDBC are classified into four types depending on the architectural relationship between the application and the data source:

1. Type I (bridges) This type of driver translates JDBC function calls into function calls of another API that is not native to the DBMS. An example is an ODBC-JDBC bridge. In this case the application loads only one driver, namely the bridge.

2. Type II (direct translation to the native API) This driver translates JDBC function calls directly into method invocations of the API of one specific data source. The driver is dynamically linked, and is specific to the data source.

3. Type III (network bridges) The driver talks over a network to a middle-ware server that translates the JDBC requests into DBMS-specific method invocations. In this case, the driver on the client site (i.e., the network bridge) is not DBMS-specific.

4. Type IV (direct translation over sockets) Instead of calling the DBMS API directly, the driver communicates with the DBMS through Java sockets. In this case the driver on the client side is DBMS-specific

With an example, explain stored procedures In SQL. **(05 Marks)**
Database Stored Procedures
Database program modules—procedures or functions—that are stored and executed by the DBMS at the database server. These are historically known as database stored procedures, although they can be functions or procedures. The term used in the SQL standard for stored procedures is persistent stored modules because these programs are stored persistently by the DBMS, similarly to the persistent data stored
Stored procedures are useful in the following circumstances:

- If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. This reduces duplication of effort and improves software modularity.
- Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.
- These procedures can enhance the modeling power provided by views by allowing more complex types of derived data to be made available to the database users. Additionally, they can be used to check for complex constraints that are beyond the specification power of assertions and triggers

```
CREATE PROCEDURE <procedure name> (<parameters>)
<local declarations>
<procedure body> ;
Example-CREATE FUNCTION Dept_size(IN deptno INTEGER)
RETURNS VARCHAR [7]
DECLARE No_of_emps INTEGER ;
SELECT COUNT(*) INTO No_of_emps
FROM EMPLOYEE WHERE Dno = deptno ;
IF No_of_emps > 100 THEN RETURN "HUGE"
ELSEIF No_of_emps > 25 THEN RETURN "LARGE"
ELSEIF No_of_emps > 10 THEN RETURN "MEDIUM"
ELSE RETURN "SMALL"
END IF ;
```

Assertions:-

It is a statement in SQL that ensures a certain condition will always exist in the database.

- Each assertion is given a constraint name and is specified via a condition similar to the where clause of an SQL query.
- For example, to specify the constraints that the salary of an employee must not be greater than the salary of the manger of the department that the employee works for in SQL, we can write the following as section.

```
create Assertion salary – constraint
Check (Not Exists ( select *
from employee E, employee M,
Department D
Where E. salary > M salary
And E.Dno  = D. D number
And D. µgr - ssn = M. ssn)) ;
```

- The constraint name salary - constraint is followed by the keyword check, which is followed by a condition in parentheses that must hold true on every database state for the assertion to be satisfied.
- The constraint name can be used later to refer to the constraint or to modify or drop it.
- Whenever some tuples in the database cause the conditions of an Assertion statement to evaluate to False, the constraint is violated.

What is a view? Explain how to create the view and how view can be dropped? What problems are associated with updating views? (08 marks)

A view is a single table that is derived from other tables. The other tables can be *base tables* or previously defined views.

In SQL, the command to specify a view is **CREATE VIEW**. The view is given a (virtual) table name (or view name), a list of attribute names, and a query to specify the contents of the view.

```
CREATE VIEW DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT Dname, COUNT (*), SUM (Salary)
FROM DEPARTMENT, EMPLOYEE
 WHERE Dnumber=Dno
 GROUP BY Dname;
```

The **DROP VIEW** command is used to dispose the view.

For example, to get rid of the view V1, we can use the SQL statement in V1A:

V1A: DROP VIEW WORKS_ON1;

Updating of views is complicated and can be ambiguous. An update on a view defined on a *single table* without any *aggregate functions* can be mapped to an update on the underlying base table under certain conditions.

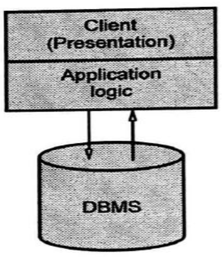
If the view involves joins, an update operation may be mapped to update operations on the underlying base relations in *multiple ways*. Hence, it is often not possible for the DBMS to determine which of the updates is intended.

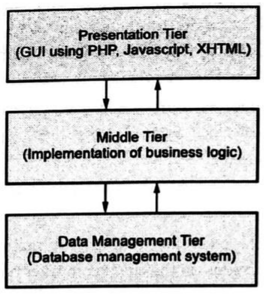
Example: Consider the WORKS_ON1 view, and suppose that the command to update the PNAME attribute of 'John Smith' is issued from 'ProductX' to 'ProductY'. This view update is shown below:

```
UPDATEWORKS_ON1
SET Pname = 'ProductY'
WHERE Lname='Smith' AND Fname='John'
AND Pname='ProductX';
```

Single Tier Architecture

- In this architecture application logic, presentation and data management all are combined in a single tier. For running the application, there was a use of mainframes. The application was accessible by **dumb terminals** that could perform only data input and display. It is as shown by following figure -

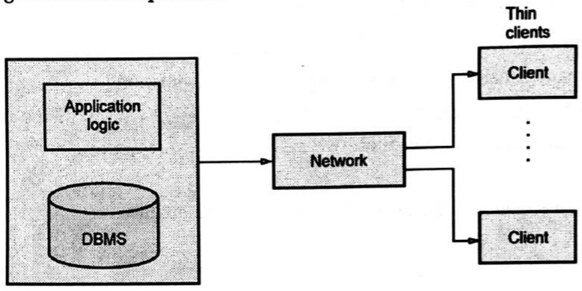




Presentation Tier
(GUI using PHP, Javascript, XHTML)
Middle Tier
(Implementation of business logic)
Data Management Tier
(Database management system)

Client Server Architecture

- The client server architecture is also called as **two tier architecture.**
- It consists of **Client computer** and **Server computer** which can communicate through well -defined protocol.



- In traditional client server architecture, the client computer implements simply the graphical user interface part. The Server computer implements application logic and data management part. In such architecture, the client is called as **thin client.**
- On the other hand there is some client server architecture, in which client implements graphical interface as well as business (application) logic part. Only data management part is taken care by server. Such client is called as **thick client.**

The three tier architecture is made up of three tiers as –

(1) **Presentation Tier :** The presentation tier is comprised of graphical user interface. The user always expects the GUI which is easy to input. He/She expects the results in some organized format. The use web-based interfaces are getting popular.

(2) **Middle Tier :** This is a tier in which the application or business logic executes. The complex business processes get executed at this tier. The business logic can be implemented in some suitable programming language like C++ or Java.

(3) **Data Management Tier :** This tier takes care of data management activities. The database management (DBMS) systems are located in this architecture.

The typical three tier architecture is represented by Fig. 4.10.3.

Advantages of Three Tier Architecture

The Three tier architecture has following advantages –

(1) **Heterogeneous Systems :** Different platforms and different software components can be used at different tiers. Also it is easy to replace or modify some code present at any tier without affecting the other code.

(2) **Thin Clients :** Clients need enough computation power for presentation layer.

How are triggers defined in SQL? Explain with example. **(05 Marks)**

A trigger is a procedure that is automatically invoked by the DBMS in response to specified changes to the database, and is typically specified by the DBA. A database that has a set of associated triggers is called an active database. A trigger description contains three parts:

Event: A change to the database that activates the trigger.

Condition: A query or test that is run when the trigger is activated.

Action: A procedure that is executed when the trigger is activated and its condition is true.

A trigger can be thought of as a 'daemon' that monitors a database, and is executed when the database is modified in a way that matches the event specification. An insert, delete or update statement could activate a trigger, regardless of which user or application invoked the activating statement; users may not even be aware that a trigger was executed as a side effect of their program.

```
CREATE TRIGGER init.count BEFORE INSERT ON Students.   /* Event */
DECLARE
count INTEGER;
BEGIN                                        /* Action */
count := 0;
END
```

(3) Integrated Data Access : In many applications, data must be accessed from several sources. This is can be done transparently in three tier architecture by using the middle tier.

(4) Scalability to Many Clients : Multiple clients can access the system through middle tier.

Briefly explain the Properties of Cursors? **(8marks)**

The general form of a cursor declaration is:

```
DECLARE cursorname [INSENSITIVE] [SCROLL] CURSOR [WITH HOLD]
        FOR some query [ ORDER BY order-item-list ]
        [ FOR READ ONLY I FOR UPDATE ]
```

- A cursor can be declared to be a read-only cursor (FOR READ ONLY) or, if it is a cursor on a base relation or an updatable view, to be an updatable cursor (FOR UPDATE).
- The UPDATE andDELETE commands allow us to update or delete the row on which the cursor is positioned.

For example, if *sinfa* is an updatable cursor and open, the following statement is executed:

```
        UPDATE Sailors S
        SET S.rating = S.rating ~ 1
        WHERE CURRENT of sinfo;
```

- If the keyword SCROLL is specified, the cursor is scrollable, which means that variants of the FETCH command can be used to position the cursor in very flexible ways; otherwise, only the basic FETCH command, which retrieves the next row, is allowed.
- If the keyword INSENSITIVE is specified, the cursor behaves as if it is ranging over a private copy of the collection of answer rows. Otherwise, and by default, other actions of some transaction could modify these rows, creating unpredictable behavior.

For example, while rows is fetched using the *sinfa* cursor, then *rating* values is modified in Sailor rows by concurrently executing the command:

```
        UPDATE Sailors S
        SET S.rating = S.rating -
```

(5) Software Development Benefits : By separating presentation, business logic and data management activities by means of separate tiers, it is easy to debug, maintain and modify the system as per the requirements. Interaction between tiers occur through well defined standardized APIs (Application Programming interfaces). Hence it is possible to create reusable components using three tier architecture.